BAB II

LANDASAN TEORI

2.1 Pengembangan Piranti Lunak

Pengembangan piranti lunak merupakan tugas yang sangat kompleks. Hal tersebut membutuhkan komunikasi dengan *customers*, mendefinisikan tugas dan hubungan antara keduanya, membuat prediksi dan sebagainya. Dengan kata lain, yaitu sebuah rencana untuk mengembangkan sebuah piranti lunak (Zima, 2015)

2.2 PEMILU (Pemilihan Umum)

(PKPU 4, 2019) Pemilihan Umum (Pemilu) adalah sarana pelaksanaan kedaulatan rakyat untuk memilih anggota Dewan Perwakilan Rakyat, anggota Dewan Perwakilan Daerah, Presiden dan Wakil Presiden, dan untuk memilih anggota Dewan Perwakilan Rakyat Daerah, yang dilaksanakan secara langsung, umum, bebas, rahasia, jujur, dan adil dalam Negara Kesatuan Republik Indonesia berdasarkan Pancasila dan Undang- Undang Dasar Negara Republik Indonesia Tahun 1945.

2.3 Dokumen-dokumen PEMILU (Pemilihan Umum)

Menggunakan sumber *file* PKPU 3 dan 4 Tahun 2019 Terdapat empat jenis dokumen pendukung yang menjadi acuan untuk dijadikan *template* pengisian data rekapitulasi suara pemilu, yaitu Berita Acara Rekapitulasi Hasil Perolehan Penghitungan Suara di tingkat TPS (Tempat Pemungutan Suara) dalam formulir Model C1, Berita Acara Rekapitulasi Hasil Perolehan Penghitungan Suara di tingkat kelurahan dalam formulir Model DAA1, Berita Acara Rekapitulasi Hasil Perolehan Penghitungan Suara di tingkat kecamatan dari masing-masing kelurahan dalam formulir Model DA1, Berita Acara Rekapitulasi Hasil Perolehan

Penghitungan Suara di tingkat kabupaten/kota dalam formulir Model DB1, Berita Acara Rekapitulasi Hasil Perolehan Penghitungan Suara di tingkat provinsi dalam formulir Model DC1. Dokumen pendukung tersebut memiliki beberapa model *form* untuk di isi oleh pengawas dari masing-masing tingkat yaitu sebagai berikut.

a. PPWP yang merupakan catatan hasil perhitungan suara pasangan calon presiden dan wakil presiden pada pemilihan umum.
b. DPR yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan rakyat pada pemilihan umum.
c. DPRA yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan rakyat Aceh pada pemilihan umum.
d. DPRP yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan rakyat Papua pada pemilihan umum.
e. DPRPB yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan rakyat Papua Barat pada pemilihan umum.
f. DPRK yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan rakyat kabupaten/kota pada pemilihan umum.
g. DPD yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan daerah pada pemilihan umum.
h. DPRD KAB/KOTA yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan rakyat daerah kabupaten/kota pada pemilihan umum.
i. DPRD PROVINSI yang merupakan catatan hasil perhitungan suara calon anggota dewan perwakilan rakyat daerah provinsi pada pemilihan umum.

2.4 Algoritma Boyer Moore

Algoritma Boyer Moore pertama kali diperkenalkan oleh Bob Boyer dan J.S. Moore pada tahun 1977. Metode ini digunakan dengan cara pencocokan kata dimulai dari karakter terakhir II-7 kata kunci menuju karakter awalnya. Jika terjadi perbedaan antara karakter terakhir kata kunci dengan kata yang dicocokkan, maka karakter-karakter dalam potongan kata yang dicocokkan tadi akan diperiksa satu per satu. Hal ini dimaksudkan untuk mendeteksi apakah ada karakter dalam potongan kata tersebut yang sama dengan karakter yang ada pada kata kunci.

Algoritma Boyer Moore termasuk algoritma *string matching* yang paling efisien dibandingkan algoritma-algoritma *string matching* lainnya. Karena sifatnya yang efisien, banyak dikembangkan algoritma *string matching* dengan bertumpu pada konsep algoritma Boyer Moore, beberapa di antaranya adalah algoritma Turbo BM dan algoritma Quick Search. Algoritma Boyer Moore menggunakan metode pencocokan string dari kanan ke kiri yaitu men-scan karakter *pattern* dari kanan ke kiri dimulai dari karakter paling kanan. Algoritma Boyer Moore menggunakan dua fungsi *shift* yaitu *good-suffix shift* dan *bad-character shift* untuk mengambil langkah berikutnya setelah terjadi ketidakcocokan antara karakter *pattern* dan karakter teks yang dicocokkan (Sibalta, 2014).

Tabel untuk penggeseran bad-character dan good-suffix dapat dihitung dengan kompleksitas waktu dan ruang sebesar $O(n + \sigma)$ dengan σ adalah besar ruang alfabet. Sedangkan pada fase pencarian, algoritma ini membutuhkan waktu sebesar $O(mn)$, pada kasus terburuk, algoritma ini akan melakukan $3n$ pencocokkan karakter, namun pada performa terbaiknya algoritma ini hanya akan melakukan $O(m/n)$ pencocokkan. Dalam proses pencarian string terdapat acuan untuk

melakukan pergeseran dalam pencarian *pattern* yaitu dengan membuat tabel *Occurrence Heuristic (OH)* dengan formula Values = Length of pattern – index - 1. Dalam pembuatan sistem informasi, boyer moore digunakan untuk mencocokkan *string* antara *file* yang di *upload* dengan modul, yang didapat pada saat meng-*upload* data ke dalam sistem informasi.

2.5 Cara kerja algoritma Boyer Moore

Algoritma Boyer Moore adalah Algoritma pencocokan *string* yang juga terdiri dari dua komponen utama, yaitu *pattern* dan teks, dan untuk penentuan *pattern* telah ditentukan di dalam modul dan teks didapat dari dokumen yang di-*upload* melalui sistem. Berikut merupakan contoh penggunaan algoritma Boyer Moore dalam melakukan pencarian dalam teks (Alwin Fau, 2017):

Teks (S) : PENGOLAHAN CITRA DIGITAL

Pattern (P) : CITRA

Tahapan pencarian *pattern* (P) dalam Teks (S) dapat dilihat pada Tabel:

Tabel 2.1 Analisa Penentuan Nilai OH dan MH

Pattern (P):	C	I	T	R	A
Occurrence Heuristic (OH) :	4	3	2	1	0
Match Heuristic (MH) :	5	5	5	5	1

Langkah-langkah:

a. Pada pergeseran pertama karakter “A” pada *pattern* tidak cocok dengan karakter “O” pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter O tidak terdapat pada tabel, sehingga pergeseran selanjutnya adalah sebanyak jumlah karakter “A” pada *pattern* yaitu 5. Langkah

ke-1 dapat dilihat pada Gambar 2.1

Langkah Ke-1																								
Posisi Teks (S):	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S):	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P):	C	I	T	R	A																			

Gambar 2.1 Langkah Ke-1 Cara Kerja Algoritma Boyer Moore

b. Pada pergeseran Ke-2 karakter "A" pada *pattern* tidak cocok dengan karakter "N" pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter N tidak terdapat pada tabel, sehingga pergeseran selanjutnya adalah sebanyak jumlah karakter "A" pada *pattern* yaitu 5. Langkah ke-2 dapat dilihat pada Gambar 2.2

Langkah Ke-2																								
Posisi Teks (S):	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S):	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P):						C	I	T	R	A														

Gambar 2.2 Langkah Ke-2 Cara Kerja Algoritma Boyer Moore

c. Pada pergeseran Ke-3 karakter "A" pada *pattern* tidak cocok dengan karakter "R" pada teks, maka pergeseran selanjutnya berdasarkan nilai dari tabel OH. Pada tabel OH karakter R terdapat pada tabel, sehingga pergeseran selanjutnya adalah sebanyak jumlah karakter "R" pada *pattern* yaitu 1. Langkah ke-3 dapat dilihat pada Gambar 2.3

Langkah Ke-3																								
Posisi Teks (S):	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S):	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P):											C	I	T	R	A									

Gambar 2.3 Langkah Ke-3 Cara Kerja Algoritma Boyer Moore

d. Pada pergeseran Ke-4 karakter "A" pada *pattern* cocok dengan karakter "A" pada teks, maka pergeseran selanjutnya dimundurkan satu langkah.

Langkah Ke-4																								
Posisi Teks (S):	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Teks (S):	P	E	N	G	O	L	A	H	A	N		C	I	T	R	A		D	I	G	I	T	A	L
Pattern (P):												C	I	T	R	A								

Gambar 2.4 Langkah Ke-4 Cara Kerja Algoritma Boyer Moore

e. Pada pergeseran selanjutnya dilakukan sampai pada pergeseran ke-12 karakter “C” pada *pattern* dengan karakter “C” pada teks cocok.

2.6 Pseudocode Algoritma Boyer Moore

Berikut merupakan pseudocode dari algoritma Boyer Moore.

```
procedure preBmBc(input x: array of char, input
m:integer, input/output bmBc: array of integer)

Deklarasi
  i:integer

Algoritma
  for (i=0; i<ASIZE; ++i)
    bmBc[i] ← m
    for (i=0; i < m-1; ++i)
      bmBc[x[i]] ← m-i-1

procedure suffixes(input x: array of char, input m:
integer, input/output suff: array of integer)

Deklarasi
   f, g, i : integer

Algoritma
  suff[m-1] ← m;
  g ← m-1

  for (i = m-2; i >= 0; --i)
    if (i>g and suff[i+m-1-f]<i-g)
      suff[i] ← suff[i+m-1-f]
    else {
      if (i<g)
        g ← i
      f=i
      while(g> 0 and x[g]=x[g+m-1-f])
        --g
      suff[i] ← f-g

procedure preBmGs(input x: array of char, input
m:integer, input/output bmGs: array of integer)

Deklarasi
  i, j : integer
  suff : array [0..XSIZE] of integer
```

Algoritma

```
  suffixes(x, m, suff)

  for (i = 0; i < m; ++i)
    bmGs[i] ← m
  j ← 0
  for (i = m-1; i >= -1; --i)
    if (i = -1 or suff[i] = i+1)
      for (j = 0; j < m-1-i; ++j)
        if (bmGs[j] = m)
          bmGs[j] ← m-1-i
  for (i = 0; i <= m-2; ++i)
    bmGs[m-1-suff[i]] ← m-1-i
```

```
procedure BM(input x: array of char, input m:integer,
input y: array of
char, input n: integer)
```

Deklarasi

```
   i, j : integer
   bmGs : array [0..XSIZE] of integer
   bmBc : array [0..ASIZE] of integer
```

Algoritma

```
   /* Preprocessing */
   preBmGs(x, m, bmGs)
   preBmBc(x, m, bmBc)

  /* Searching */
  j=0
  while (j <= n-m)
    for (i = m-1; i >= 0 and x[i] = y[i+j]; --i)
      if (i < 0)
        OUTPUT(j)
        j ← j + bmGs[0]
      else
        j ← j + MAX(bmGs[i], bmBc[y[i+j]]-m+1+i)
```

2.7 USE Questionnaire

Penelitian ini menggunakan kuesioner USE menjadi alat ukur untuk mendapatkan nilai *usability*. (Ningrum, 2019) Terdapat 3 aspek pengukuran *usability* pada kuesioner ini, yaitu *learnability*, *efficient* dan *satisfaction*. Masih ada beberapa parameter selain 3 aspek tersebut, tetapi 3 aspek itu yang lebih mudah

untuk diamati dan dibandingkan hasilnya. Kuesioner USE memiliki 30 pernyataan yang terbagi menjadi 4 parameter. Setiap pertanyaan yang mewakili penilaian kegunaan saat pengguna menggunakan aplikasi tersebut.

2.8 Skala Likert

Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam kuesioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei. Sewaktu menanggapi pertanyaan dalam skala Likert, responden menentukan persetujuan mereka terhadap suatu pernyataan dengan memilih salah satu dari pilihan yang tersedia (Syofian, 2015). Biasanya disediakan lima pilihan skala dengan format seperti:

1. Sangat Setuju
2. Setuju
3. Netral
4. Tidak setuju
5. Sangat Tidak Setuju